INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computing deep facet-defining disjunctive cuts for mixed-integer programming

Florent Cadoux

N° 6177

May 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6177--FR+ENG

Computing deep facet-defining disjunctive cuts for mixed-integer programming

Florent Cadoux*

Thème NUM — Systèmes numériques
Projet Bipop

Rapport de recherche n° 6177 — May 2007 — 22 pages

Abstract: The problem of separation is to find an affine hyperplane, or "cut", that lies between the origin O and a given closed convex set Q in a Euclidean space. We focus on cuts which are deep for the Euclidean distance, and facet-defining. The existence of a unique deepest cut is shown and cases when it is decomposable as a combination of facet-defining cuts are characterized using the reverse polar set. When Q is a split polyhedron, a new description of the reverse polar is given. A theoretical successive projections algorithm is proposed that could be used to compute deep facet-defining split cuts.

Key-words: integer programming, separation, cut generation, disjunctive cut, convex analysis, reverse polar

* florent.cadoux@inria.fr

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Coupes disjonctives profondes exposant des facettes pour la programmation entière générale

Résumé : Le problème de séparation consiste à trouver un hyperplan affine, ou "coupe", situé entre l'origine O et un ensemble convexe fermé Q dans un espace euclidien. On s'intéresse aux coupes profondes au sens de la distance euclidienne, et qui exposent une facette. L'existence d'une unique coupe de profondeur maximale est prouvée, et les cas où elle peut être décomposée en combinaison de coupes exposant une facette sont caractérisés grâce au polaire inverse de Q. Quand Q est un polyèdre disjonctif, une nouvelle description du polaire inverse est donnée. Un algorithme théorique de projections successives est proposé, qui pourrait être utilisé pour calculer des coupes profondes exposant une facette.

Mots-clés : programmation entière, séparation, génération de coupes, coupes disjonctives, analyse convexe, polaire inverse

Introduction

The problem of *separation* is essential in combinatorial optimization, occuring for instance in the following context:

- a function must be optimized on a "complicated" set in $\mathbb{R}^n$, typically a set of integer points;
- this problem being too hard, an easier relaxation is solved;
- the problem is then to separate the solution of the relaxation from the complicated set, so as to tighten the relaxation.

Separating hyperplanes, called *cuts* in the community of combinatorial optimization, are used in many practical methods aimed at solving mixed integer programs, often embedded in a branch-and-bound framework like the branch-and-cut algorithm. An overview of the existing general techniques to compute such cuts can be found in [3]. We will focus on particular cuts, called *split cuts* which belong to the family of *disjunctive cuts* [1]. In [2], lift-and-project cuts are defined and shown to be a particular case of split cuts for the mixed 0-1 case. In [7], a procedure is devised to generate one facet-defining lift-and-project cut. In this paper, we address the problem of generating facet-defining cuts which are *deep*; besides, we consider the general case of split cuts for the mixed integer case.

The key idea to address the general problem of separation between O and some convex set Q is to use a precise correspondence between facets of Q and extreme points of another convex set in the dual space: the *reverse polar* of Q. Moreover, a facet is deep in some sense when the corresponding extreme point of the reverse polar is close to the origin in the dual space. These two facts lead us to a theoretical algorithm aimed at computing deep facet-defining cuts by solving an optimization problem, more precisely a quadratic programming problem, on the reverse polar set. We then particularize the theory to the special case, of great practical importance, where the point to be separated is the solution of the linear relaxation of a mixed integer program, and the convex set is a disjunctive (split) polyhedron.

The article is organized as follows: section 1 introduces our main object, the *reverse polar* of a convex set. Section 2 advocates deep, facet-defining cuts as a good choice for the applications. A characterization of such cuts is given which uses the projection of O onto the reverse polar Q^- and its decomposition as a convex combination of extreme points of Q^-. Section 3 is devoted to the problem of computing the projection of a point onto a closed convex set, and the above-mentionned decomposition. Section 4 gives a characterization of Q^- when Q is an explicitly described polyhedron, which shows that computing the projection of O onto Q^- is tractable in this case. Section 5 introduces disjunctive programming and section 6 generalizes the method of section 4 to the case where Q is a split polyhedron. We finally propose a theoretical algorithm to compute deep, facet-defining cuts split cuts for general mixed integer programs.

1 Reverse polar and normal cone of a convex set

Let us first recall a few definitions and properties to introduce our key object: the reverse polar of a convex set. Let $Q \subsetneq \mathbb{R}^n$ be a nonempty closed convex set, such that $O \notin Q$. We aim at separating O from Q. The elements of $\mathbb{R}^n$ are identified with column vectors, and the standard Euclidean scalar product of vectors x and y is denoted by $x \cdot y$. We will use the so-called *support function* $\sigma_S : \mathbb{R} \to \mathbb{R} \cup +\infty$ of a convex set $S \subset \mathbb{R}^n$:

$$\sigma_S(d) := \sup_{x \in S} d \cdot x. \tag{1}$$

For this and other elementary concepts of convex analysis such as recession cones and exposed faces, we refer to [5].

Definition (Reverse polar) Let $Q \subsetneq \mathbb{R}^n$ be a nonempty closed convex set with $O \notin Q$. The set

$$Q^- := \{d \in \mathbb{R}^n \ ; \ \sigma_Q(d) \leqslant -1\} = \{d \in \mathbb{R}^n \ ; \ \forall x \in Q \ : \ d \cdot x \leqslant -1\} \tag{2}$$

is called the *reverse polar* of Q.

This set was introduced by Balas [1] and is very convenient for cutting: indeed, d separates O from Q if and only if $d \in Q^-$ (up to multiplication by a positive constant), via the hyperplane of equation $d \cdot x = \sigma_Q(d)$.

Definition (Normal cone) Let $Q \subsetneq \mathbb{R}^n$ be a nonempty closed convex set with $O \notin Q$. The set

$$N_Q := \{d \in \mathbb{R}^n \ ; \ \sigma_Q(d) \leqslant 0\} = \{d \in \mathbb{R}^n \ ; \ \forall x \in Q \ : \ d \cdot x \leqslant 0\} \tag{3}$$

is called the *normal cone* of Q.

It is a (small) generalization of the usual notion of normal cone, since usually $O \in Q$ is required in this definition. Note that this definition is very similar to the one of Q^-: the -1 in the right hand side was simply replaced with 0. The normal cone will be useful in the description of Q^- via the following theorem.

Theorem 1.1 (Recession cone) *Let $Q \subsetneq \mathbb{R}^n$ be a nonempty closed convex set with $O \notin Q$. The recession cone of Q^- is exactly N_Q:*

$$(Q^-)_\infty = N_Q. \tag{4}$$

Proof By definition of Q^- and N_Q we have $Q^- \subset N_Q$ so that the recession cone of Q^- is a subset of $(N_Q)_\infty = N_Q$. Conversely, $\forall d_1 \in Q^-$ and $\forall d_2 \in N_Q$, we have : $\forall t \geqslant 0$, $\forall x \in Q$, $(d_1 + t d_2) \cdot (x - q) \leqslant -1$ so $d_1 + t d_2 \in Q^-$, which implies that $d_2 \in (Q^-)_\infty$. ∎

The case when Q is (the closed convex hull of) the union of two convex sets will appear in the sequel, therefore we now recall the following result, taken from [4].

Theorem 1.2 (Reverse polar of a union) *Let Q_0 and Q_1 be two nonempty closed convex sets in $\mathbb{R}^n$, with $O \notin Q := \overline{conv}(Q_0 \cup Q_1)$. There holds*

$$Q^- = Q_0^- \cap Q_1^-. \tag{5}$$

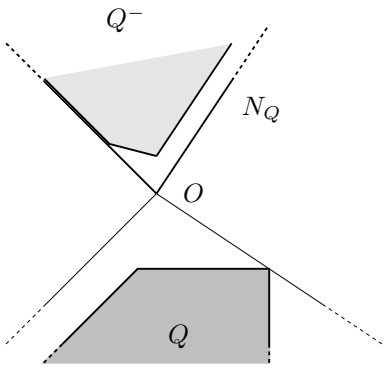

Figure 1: Normal cone and reverse polar of a polyhedron

2 "Good" cuts

There are infinitely many possible cuts, and this section is devoted to choosing good ones. A first natural requirement for a cut is to touch Q. This section introduces two more requirements: to be deep and facet-defining.

2.1 Facet-defining cuts

The particular case where Q is a polyhedron deserves special attention. Recall that by definition, a *facet* of a polyhedron Q is a face (that is, the intersection of Q with some affine hyperplane supporting it) of maximal dimension, distinct from Q. When Q is a polyhedron, the inequalities that define a facet of Q play a particular role, as stated by the following theorem taken from chapter 8 in [8].

Theorem 2.1 *Let $Q \subset \mathbb{R}^n$ be a full-dimensional polyhedron described by irredundant inequalities $Ax \leqslant b$. Then $Ax \leqslant b$ is the unique minimal representation of Q, up to multiplication of the inequalities by positive scalars. Moreover, there exists a one-to-one correspondence between the facets of Q and the inequalities in $Ax \leqslant b$.*

As a consequence, when Q is a polyhedron, we are especially interested in cuts which expose a facet of Q. This motivates the next result (Theorem 6.2 of [4]).

Theorem 2.2 *Let $Q \subsetneq \mathbb{R}^n$ be a polyhedron such that $lin(Q) = \mathbb{R}^n$ and $O \notin Q$. Then Q^- is a polyhedron, and the face of Q exposed by $d \in Q^-$ is a facet of Q if and only if*

$$d^* := -\frac{d}{\sigma_Q(d)} \tag{6}$$

is an extreme point of Q^-.

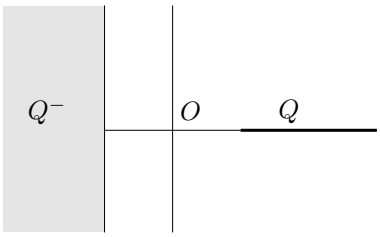

Figure 2: When $lin(Q) \subsetneq \mathbb{R}^n$ ...

Remark Theorem 2.2 does not hold without the assumption that $lin(Q) = \mathbb{R}^n$. For instance on Figure 2, in two dimensions, the linear hull $lin(Q)$ is the x axis which is not full-dimensional. The orthogonal direction $Q^\perp$ is the y axis and Q^- is invariant along the y axis, therefore having no extreme point. Theoretically, this problem can be avoided by working in the subspace $lin(Q)$ instead of $\mathbb{R}^n$, but in practice we may not know anything about $lin(Q)$. This problem will be dealt with in section 6.

2.2 Deep cuts

An idea supported in [4] is to consider additionally the *depth* of the cut, defined as the Euclidean distance from O to the cutting hyperplane. The following theorem states that there exists a deepest cut. It touches Q at the projection q^* of O onto the convex Q.

Theorem 2.3 *Let $Q \subsetneq \mathbb{R}^n$ be a nonempty convex set with $O \notin Q$, q^* the projection of O onto the convex set Q and $H = \{x \in \mathbb{R}^n \ : \ d \cdot x = \beta\}$ an hyperplane that separates O from Q. Let H' be the hyperplane orthogonal to q^* that contains q^*. There holds*

$$d(O,H) \leqslant d(O,H') \quad \text{and} \quad d(O,H) = d(O,H') \Longleftrightarrow H = H'. \tag{7}$$

Proof Let h be the projection of O onto H and q' the point on the line segment $[O,q^*]$ which is contained in H (its existence is ensured by the Theorem of intermediate values, and uniqueness holds since $O \notin H$). There holds

$$d(O,H) = d(O,h) \leqslant d(O,q') \leqslant d(O,q^*) = d(O,H')$$

and the inequalities are equalities if and only if $h = q'$ (by uniqueness of the projection on H) and $q' = q^*$ (O, q' and q^* are aligned), that is to say, if and only if $H = H'$. ∎

This result reveals the importance of the projection onto a closed convex set C ($C = Q$ in Theorem 2.3). Recall the following characterization of a projection.

Theorem 2.4 *Let C be a closed convex set and $c^* \in C$. Then c^* is the projection of the origin onto C if and only if $c^* \cdot c \geqslant \|c^*\|^2$ for any $c \in C$.* ∎

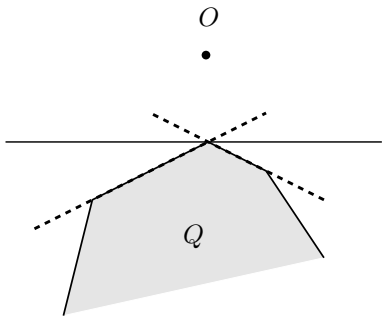

Figure 3: Deepest cut and facets of Q

2.3 Deep facet-defining cuts

The two properties introduced above, to be deep and facet-defining, are antagonistic. For example, Figure 3 clearly shows that the deepest cut (in full line) does not expose a facet; and the two facet-defining cuts (in dashed line) are moderately deep. On the other hand, they *imply* the deepest cut.

At this point, it is quite natural to look for such cuts, which expose facets of Q, and which imply the deepest cut as a consequence. This question can be more conveniently formulated in the dual space, thanks to the next result.

Theorem 2.5 *Let q^* be the projection of O onto Q and d^* its projection onto Q^-. There holds*

$$d^* = -\frac{q^*}{|q^*|^2}. \tag{8}$$

Proof By definition, for all $d \in Q^-$,

$$-1 \geqslant \sigma_Q(d) \geqslant d \cdot q^* \geqslant -\|d\| \, \|q^*\| \, ;$$

so we have proved

$$\|d\| \geqslant \frac{1}{\|q^*\|} \, , \quad \text{for all } d \in Q^- \ . \tag{9}$$

Now set $d^* := -\frac{q^*}{|q^*|^2}$: for all $x \in Q$,

$$d^* \cdot x = -\frac{q^* \cdot (x - q^* + q^*)}{\|q^*\|^2} = -\frac{q^* \cdot (x - q^*)}{\|q^*\|^2} - 1;$$

but $q^* \cdot (x - q^*) \geqslant 0$ (Theorem 2.4). So we have proved

$$d^* \cdot x \leqslant -1\,, \quad \text{for all } x \in Q$$

hence $d^* \in Q^-$. Since obviously $|d^*| = \frac{1}{|q^*|}$, (9) establishes that d^* has shortest norm in Q^-. ∎

This result is important with relation to our previous discussion: computing facet-defining cuts implying the deepest cut amounts to computing extreme points of Q^- having the projection d^* as a convex combination. Indeed, consider a set of hyperplanes defined by the equation $d_k \cdot x = \sigma_Q(d_k)$; each of them separates 0 from Q if $\sigma_Q(d_k) < 0$, and defines a facet of Q if d_k is an extreme point of Q^- (Theorem 2.2; then $\sigma_Q(d_k) = -1$); and they altogether imply the deepest cut if $q^* = -\sum_k \alpha_k d_k$, for some coefficients $\alpha_k \geq 0$ (Theorem 2.3). The above result says that this latter property means that $\sum_k \alpha_k d_k$ is collinear to d^*; we may assume that the α_k sum up to 1, so that $d^* = \sum_k \alpha_k d_k$.

3 Projection algorithm

The previous section showed the importance of minimizing the quadratic function $\|\cdot\|^2$ (the square of the Euclidean distance) in $\mathbb{R}^n$ to compute deep cuts. It also showed the importance of decomposing the projection in extreme points. In this section, we present an algorithm to solve this problem where the extreme points are generated by a sequence of linear programs. In section 6 we will see a situation where these linear programs are tractable.

3.1 The algorithm

Let C be a closed bounded convex set. The following algorithm computes the projection of the origin onto C by a *column-generation* mechanism, illustrated by Fig. 4. Note that, being bounded, C has an extreme point so that the initialization makes sense.

Algorithm 1 (Projection by column generation) *Choose an extreme point d_1 of C; set $k = 1$.*

Step 1 *(*Master problem*). Compute the projection d_k^* of the origin onto the convex hull of $d_1, \ldots, d_k$.*

Step 2 *(*Oracle call*). Compute an extreme point $d_{k+1} \in C$ by minimizing over C the linear function $d \mapsto d_k^* \cdot d$.*

Step 3 *(*Stopping test*). If $d_k^* \cdot d_{k+1} < \|d_k^*\|^2$, increase k by 1 and loop to Step 1. Otherwise stop.* ∎

Step 1 consists in solving for $\alpha \in \mathbb{R}^k$

$$\min \frac{1}{2}\Big\|\sum_{i=1}^{k} \alpha_i d_i\Big\|^2\,, \quad \sum_{i=1}^{k} \alpha_i = 1\,, \quad \alpha_i \geqslant 0,\ i = 1, \ldots, k\,. \tag{10}$$

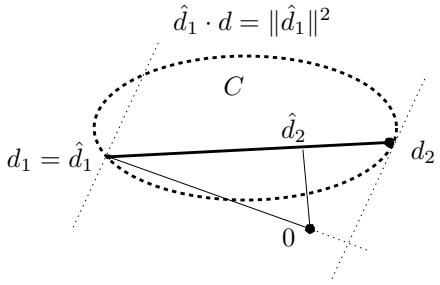

Figure 4: The first two iterations of algorithm 1

Any standard QP algorithm can be used to solve this standard QP problem; Ph. Wolfe described one in [9], adapted to the particular form (10). The output of any such algorithm is a set of positive α_i's; the corresponding d_i's are extreme points of C (by construction in Step 2).

Algorithm 1 will become completely defined and implementable if, in addition to a QP solver, an oracle is available to minimize a linear function over C. This oracle must always return extreme points.

Remark In its original formulation [9], Wolfe's algorithm explicitly uses the list of points (here $d_1, \ldots, d_k$) onto wich the projection is computed. This list is scanned whenever the algorithm needs the smallest of k numbers of the form $X \cdot d_1, \ldots, X \cdot d_k$.

Instead of repeated calls to a QP solver, a variant of Algorihm 1 could call Wolfe's algorithm just once, to project directly the origin onto C. Instead of the (unknown) list of extreme points of C, Wolfe's algorithm would be input with the above-mentioned oracle, to minimize $X \cdot d$ over C. ∎

If the algorithm stops at some iteration K, then Steps 2 and 3 imply that $d_K^* \in C$ satisfies $d_K^* \cdot d \geqslant \|d_K^*\|^2$ for all $d \in C$; in view of Theorem 2.4, this exactly means that d_K^* is the projection of the origin onto C.

Now suppose Algorithm 1 were applied to $C = Q^-$ and did stop at some iteration K. In the language of the present paper, d_K^* would be the deepest cut (Theorem 2.5). More importantly, a number of cuts d_i with $\alpha_i > 0$ would be produced, which would be facet-defining (Theorem 2.2); because $d_K^* = \sum_i \alpha_i d_i$, they would imply the deepest cut as a consequence. In a word, Algorithm 1 would do just the job mentioned in section 2. However, the identification $C = Q^-$ violates the inherently unbounded character of Q^-; subsection 3.3 will address this difficulty.

3.2 Convergence

The d_{k+1} produced by Step 2 of Algorithm 1 cannot be any of the previous d_i's; the algorithm is therefore finite if C has finitely many extreme points. However, a more intrinsic conver-

gence proof points out the role of boundedness of C and lays the ground for a complexity study.

Lemma 3.1 *Setting $d(t) := d_k^* + t(d_{k+1} - d_k^*)$, there holds for all $t \geqslant 0$*

$$\|d(t)\|^2 \leqslant \|d_k^*\|^2 - 2t\|d^* - d_k^*\|^2 + t^2\|d_{k+1} - d_k^*\|^2 .$$

Proof In the development

$$\|d(t)\|^2 = \|d_k^*\|^2 + 2td_k^* \cdot (d_{k+1} - d_k^*) + t^2\|d_{k+1} - d_k^*\|^2 ,$$

we work out the coefficient of $t \geqslant 0$: from the definition of d_{k+1} in Step 2 ($d^* \in C$!):

$$\begin{array}{rcl} d_k^* \cdot (d_{k+1} - d_k^*) & \leqslant & d_k^* \cdot (d^* - d_k^*) \\ & = & d^* \cdot (d^* - d_k^*) - \|d_k^* - d^*\|^2 \\ & \leqslant & -\|d_k^* - d^*\|^2 , \end{array}$$

where the last inequality comes from Theorem 2.4 ($d_k^* \in C$!). ∎

Theorem 3.2 *The sequence $\{d_k^*\}$ converges to d^*.*

Proof Because d_{k+1} and d_k^* vary in the bounded set C, $\|d_{k+1} - d_k^*\|^2 \leqslant M$ for some number M. Now take $\delta > 0$ and call K_δ the set of k such that $\|d^* - d_k^*\|^2 > \delta$. Diminishing δ if necessary, we may assume $\delta/M \leqslant 1$; then $d(\delta/M)$ lies in the segment $[d_k^*, d_{k+1}]$ and, from Step 1:

$$\|d_{k+1}^*\|^2 \leqslant \|d(t)\|^2 \leqslant \|d_k^*\|^2 - 2t\delta + t^2M$$

for all $t \in [0, \delta/M]$. Setting in particular $t = \delta/M$:

$$\|d_{k+1}^*\|^2 \leqslant \|d_k^*\|^2 - \frac{\delta^2}{M} , \quad \text{for all } k \in K_\delta .$$

The whole sequence $\{\|d_k^*\|^2\}$ is decreasing, so the above inequality can hold for finitely many k only: K_δ is a finite set.

In other words: for δ arbitrarily small, $\|d^* - d_k^*\|^2 \leqslant \delta$ if k is large enough; this means that $\|d_k^* - d^*\|^2 \to 0$. ∎

3.3 The boundedness assumption

When C is unbounded, the proof of Theorem 3.2 suggests that algorithm 1 may not converge. Moreover, if C is unbounded, linear functions may be unbounded from below: Step 2 may produce no d_{k+1}.

Indeed, the projection d^* need not be a convex combination of extreme points: extreme rays may be necessary to describe it. In our present framework, this corresponds to a deepest cut impossible to describe by facet-defining cuts; see Fig. 5. In the unbounded case, Algorithm 1 may therefore be trying to compute non-existing objects.

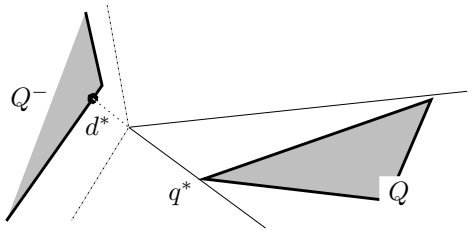

Figure 5: The deepest cut need not be implied by facet cuts

Admittedly, these arguments disappear when C is polyhedral: it can then be written

$$C = \sum_{i=1}^{p} \alpha_i d_i + \sum_{j=1}^{r} \lambda_j r_j \,, \tag{11}$$

where α varies in the unit-simplex and $\lambda \geqslant 0$. The projection d^* can thus be described as a combination of extreme points d_i and extreme rays r_j. With appropriate modifications, Algorithm 1 can still make sense in the unbounded but polyhedral case.

We note, however, that extreme rays of $C = Q^-$ play a little role in our framework (Fig. 5 reveals that they do not define facet-cuts of Q). Besides, numerical difficulties still exist, as illustrated by Fig. 6:

– At the first iteration, the oracle answers the horizontal ray r_2.
– Then d_2^* is the projection of 0 onto $d_1 + r_2$.
– Numerically, d_2^* is not exactly orthogonal to r_2 and the LP oracle called at d_2^* may very well produce r_2 again (instead of the extreme point d_2).
– From then on, Algorithm 1 starts looping.

To alleviate this difficulty, an *explicit* description of the extreme elements of C in (11) seems necessary.

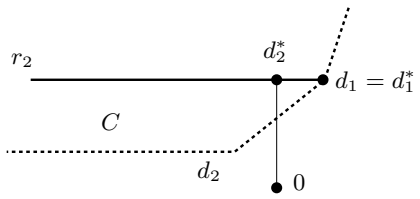

Figure 6: Numerical instablity in the unbounded case

The above reasons explain that, in our application where $C = Q^-$ is definitely unbounded, we will artificially bound Q^- by adding a normalization constraint.

4 Case of a polyhedron defined by inequalities

We assume in this section that Q is a nonempty polyhedron explicitly described by inequalities. Then we characterize the reverse polar Q^- as the envelope of its extreme points and extreme rays.

4.1 Characterization of the normal cone

The following lemma ([4], Theorem 6.7) characterizes the normal cone of Q.

Lemma 4.1 *Let $Q = \{x \in \mathbb{R}^n \ ; \ Ax \leqslant b\}$ where $m \in \mathbb{N}$, $A \in \mathcal{M}_{m,n}(\mathbb{R})$ and $b \in \mathbb{R}^m$. Assume Q nonempty and $O \notin Q$. Then the normal cone of the polyhedron Q is*

$$N_Q = \{d = A^\top u \ ; \ u \geqslant 0 \ ; \ b \cdot u \leqslant 0\}. \tag{12}$$

Said otherwise, N_Q is the image by $A^\top$ of the cone:

$$K = \{u \geqslant 0 \ ; \ b \cdot u \leqslant 0\}. \tag{13}$$

Remark Theorem 4.1 does not hold without the assumption that Q is nonempty. In practice however, we may not know whether Q is empty or not. This problem will be dealt with in section 6 and a counterexample to Theorem 4.1 will be shown when $Q = \emptyset$.

Let

$$\begin{aligned} I_- &= \{i \in 1...m \ : \ b_i < 0\} \\ I_0 &= \{i \in 1...m \ : \ b_i = 0\} \\ I_+ &= \{i \in 1...m \ : \ b_i > 0\}. \end{aligned} \tag{14}$$

The following result gives the extreme rays of K and generators of N_Q.

Theorem 4.2 *Let I_-, I_0, I_+ as in (14).*

1. *Let $(e_1, \ldots, e_m)$ be the canonical basis of $\mathbb{R}^m$. The extreme rays of the cone K are exactly (up to multiplication by a positive scalar) the elements of*

$$E = \{e_i \ ; \ i \in I_- \cup I_0\} \cup \{b_j e_i - b_i e_j \ ; \ (i,j) \in I_- \times I_+\}. \tag{15}$$

2. *The image by $A^\top$ of E generates the normal cone N_Q:*

$$cone(A^\top E) = N_Q. \tag{16}$$

Proof Inspection of (13) shows that the extreme rays of $K \subset \mathbb{R}^m$ are obtained as follows:
– Extract from $\{e_1, \ldots, e_m, b\}$ all possible sets of $m-1$ linearly independent vectors.
– Solve the corresponding systems of equations, thus producing a line in $\mathbb{R}^m$. Note that this line cannot be contained in K because $K \subset \mathbb{R}^m_+$.

– If this line has only 0 as feasible point in (13), the extreme point $0 \in K$ is produced; but no extreme ray.
– Otherwise, the feasible half of that line is an extreme ray of K.

To extract an $(m-1)$-uple, i.e. to delete a couple from $\{e_1, \ldots, e_m, b\}$, there are two possibilities.

1. Either one deletes e_i ($i \in [1, m]$) and b, then the $m-1$ remaining vectors e_k with $k \neq i$ are automatically independent, and the intersection of K with the half-line $\mathbb{R}_+ e_i$ is not the singleton $\{0\}$ if and only if $e_i \cdot b \leqslant 0$.

2. Or one deletes e_i and e_j ($i \neq j$). In this case the family is linearly independent if and only if $(e_i \cdot b, e_j \cdot b) \neq (0, 0)$ (we can even assume that $e_i \cdot b \neq 0$ and $e_j \cdot b \neq 0$ otherwise we are back to the previous case where the line is directed by one of the basis vectors), and then the line defined by these $m-1$ vectors is

$$\{u = \alpha e_i + \beta e_j \ ; \ u \cdot b = 0\}$$

A directing vector of this line is $(b \cdot e_j)e_i - (b \cdot e_i)e_j$. The intersection of K with the line is nontrivial if and only if $b \cdot e_i$ and $b \cdot e_j$ have opposite sign (we assumed they were both nonzero).

Equation (16) comes from $N_Q = A^\top K = A^\top \, cone(E) = cone(A^\top E)$. ∎

Remark Since the vectors in E contain only one or two nonzeros, the generators $A^\top E$ of N_Q computed with this method are (the transpose of) either rows of A or linear combinations of two rows of A: if A is sparse, so will be the computed generators.

Remark Note that we considered the Cartesian product of I_- and I_+ and E has *a priori* $O(m^2)$ elements. More precisely, the number of computed generators is:

$$|E| = |I_-| + |I_0| + (|I_-|)(|I_+|). \tag{17}$$

This technique thus provides us with a set of vectors containing the extreme rays of $N_Q(q)$, but possibly also some "useless" vectors which are not extreme.

Example Consider for Q the square with vertices (-3,-1) , (-3,1) , (-1,-1) , and (-1,1). Let

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{pmatrix} \text{ and } b = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 3 \end{pmatrix}$$

and since $q = (0, 0)$ there holds $w := (Aq - b) = -b = (-1, 1, -1, -3)$. Thus $I_+ = \{1, 3, 4\}$ and $I_- = \{2\}$. The extreme rays of K are the vectors $e_2 = (0, 1, 0, 0)$, $a_{2,1} = (1, 1, 0, 0)$, $a_{2,3} = (0, 1, 1, 0)$ and $a_{2,4} = (0, 3, 0, 1)$. Their image by $A^\top$ are respectively $(1, 0)$, $(1, 1)$, $(1, -1)$ and $(2, 0)$. Observe that the extreme rays of N_Q are given by the second and the third vectors ($(1, 1)$ and $(1, -1)$), whereas the two others are not extreme.

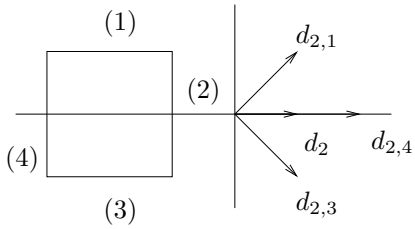

Figure 7: Generators of a normal cone

4.2 Characterization of the reverse polar

When the polyhedron Q is full dimensional and a description by inequalities of Q is available, it is easy to find the extreme points of Q^-.

Theorem 4.3 *Let $Q \subsetneq \mathbb{R}^n$ be a full dimensional polyhedron explicitly described by a set of inequalities $Ax \leqslant b$, with $O \notin Q$. Denote by a_i the rows of A, so that*

$$Ax \leqslant b \Longleftrightarrow a_i \cdot x \leqslant b_i,\ i = 1, \ldots, m.$$

1. *If the description $Ax \leqslant b$ is minimal, then the extreme points of Q^- are exactly the*

$$P_i := -\frac{a_i}{b_i} \text{ with } i \in I_-. \tag{18}$$

2. *If the description $Ax \leqslant b$ is not minimal, the P_i defined in (18) contains all the extreme points of Q^-, plus some points which lie in Q^- but are not extreme.*

Proof When the description $Ax \leqslant b$ of Q is minimal, we know the facets of Q. A direct application of Theorem 2.2 yields the result. When it is not minimal, a minimal subset of inequalities in $Ax \leqslant b$ yields the extreme points of Q^-. ∎

Remark In practice, Q may not be full dimensional. We will see in section 6 how to get rid of this problem.

We are now able to state an important theorem which characterizes Q^- when Q is a polyhedron defined by inequalities.

Theorem 4.4 *Let $Q \subsetneq \mathbb{R}^n$ be a full dimensional polyhedron explicitly described by a set of inequalities $Ax \leqslant b$, with $O \notin Q$. Let I_-, I_0, I_+ be defined by (14), the set E defined in Theorem 4.2 and the P_i for $i \in I_-$ defined in Theorem 4.3. There holds:*

$$Q^- = conv(P_i\ ;\ i \in I_-) + cone(A^\top E). \tag{19}$$

Proof Being a polyhedron, Q^- is the sum of the convex hull of its extreme points and of its recession cone $(Q^-)_\infty$. Theorem 4.3 gives the extreme points; Theorem 1.1 states that $(Q^-)_\infty = N_Q$ and Theorem 4.2 that $N_Q = cone(A^\top E)$. ∎

5 Application to disjunctive programming

The results given so far apply to the problem of *disjunctive programming*, defined in [1], with applications in integer programming. A first result is that the closed convex hull of the union of two (or more genrally, finitely many) polyhedra is a closed polyhedron. Indeed, it is the image by a linear transform (a projection) of a polyhedron in a higher dimensional space (lemma 6.4 of [4]).

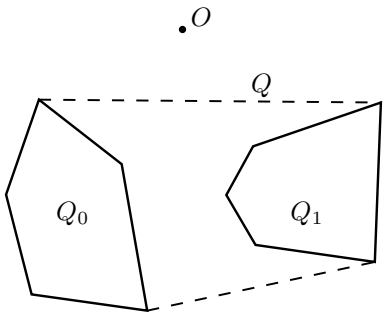

Figure 8: Disjunctive programming

Definition (Disjunctive programming) Let Q_0 and Q_1 be two nonempty closed polyhedra in $\mathbb{R}^n$, such that $O \notin \overline{conv}(Q_0 \cup Q_1)$. The polyhedron $Q = \overline{conv}(Q_0 \cup Q_1)$ is called a *disjunctive polyhedron* and a cut that separates O from Q is called a *disjunctive cut*.

The next subsection illustrates the idea of disjunctive programming. It is devoted to a particular and common type of disjunctions, the *split disjunctions*.

5.1 Split disjunctions

Consider a general *mixed integer linear program* of the form

$$\begin{cases} \min c \cdot x \\ (x_1, \ldots, x_n) \in \mathbb{Z}^n \\ (x_{n+1}, \ldots, x_{n+p}) \in \mathbb{R}^p \\ Ax \leqslant b \end{cases} \tag{20}$$

with $n, p, m \in \mathbb{N}$, $A \in \mathcal{M}_{m,n+p}(\mathbb{R})$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Practical algorithms used to solve such optimization problems often use the following relaxation of (20) :

$$\begin{cases} \min c \cdot x \\ (x_1, \ldots, x_{n+p}) \in \mathbb{R}^{n+p} \\ Ax \leqslant b. \end{cases} \tag{21}$$

which is much easier to solve. The problem of *cut generation* is to separate an optimal solution $\overline{x}$ of (21) from the feasible set in (20). Let R be the relaxed polyhedron:

$$R = \{x \in \mathbb{R}^{n+p} \ ; \ Ax \leqslant b\}$$

and let $i \in 1 \ldots n$ be such that $\overline{x}_i \notin \mathbb{Z}$ (if no such i exists, (20) is solved). Let

$$\pi = e_i \text{ and } \pi_0 = \lfloor \overline{x}_i \rfloor. \tag{22}$$

The feasible set of (20) is entirely contained in the union of the two following polyhedra

$$\begin{array}{l} Q_0 = R \cap \{x \in \mathbb{R}^n \ ; \ \pi \cdot x \leqslant \pi_0\} \\ Q_1 = R \cap \{x \in \mathbb{R}^n \ ; \ \pi \cdot x \geqslant \pi_0 + 1\} \end{array} \tag{23}$$

whereas $\overline{x}$ is not. Admitting that $\overline{x}$ is an extreme point of R, it cannot lie in $\overline{conv}(Q_0 \cup Q_1)$. To separate it from the feasible set in (20), it suffices to separate it from $\overline{conv}(Q_0 \cup Q_1)$.

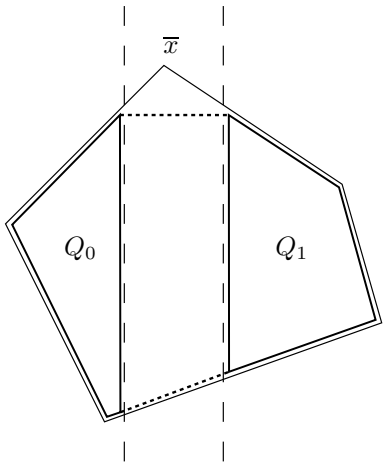

Figure 9: Particular case of a split cut

5.2 Reverse polar of a split polyhedron

Theorem 5.1 particularizes Theorem 4.4 when Q is a split polyhedron.

Theorem 5.1 *Let $R \subsetneq \mathbb{R}^n$ be a full dimensional polyhedron explicitly described by m inequalities, $\overline{x} \in R$, $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}$, Q_0 and Q_1 defined by (23) and $Q = \overline{conv}(Q_0 \cup Q_1)$. Assume that $\overline{x} \notin Q$ and $Q_0 \neq \emptyset$, $Q_1 \neq \emptyset$. Then, for $i = 0, 1$, Q_i^- is a translated polyhedral cone: $Q_i^- = P_i + K_i$, where*

$$P_0 = -\frac{\pi}{\pi_0},\ P_1 = -\frac{\pi}{\pi_0 + 1}, \tag{24}$$

and each K_i is generated by $m+1$ vectors a_i^j given by Theorem 4.4.

As a result,

$$Q^- = \bigcap_{i=0,1} (P_i + cone(a_i^j,\ j = 1 \dots m+1)) = \bigcap_{i=0,1} (\{\sum_{j=1}^{m+1} \alpha_i^j a_i^j,\ \alpha_i^j \geqslant 0\}) \tag{25}$$

Proof Note that $\overline{x}$ satisfies all the constraints defining each Q_i except the splitting inequality: I_- of (14) is a singleton, each Q_i^- has only one extreme point described by Theorem 4.3. The number of generators given by Theorem 4.2 is

$$|E| = |I_-| + |I_0| + |I_-|\,|I_+| = 1 + |I_0| + |I_+| = 1 + m. \qquad \square$$

Thus the description of $Q^- = Q_0^- \cap Q_1^-$ when Q_0 and Q_1 come from a split cut is fairly simple, since I_- is a singleton.

5.3 A particular cut

The description given in section 5.2 allows us to find a particular point of Q^-, that is, a cut. Let $\{a_i,\ i = 1 \dots n\}$ be rows of A corresponding to n independent constraints active at $\overline{x}$. Note that the a_i are among the generators of both cones Q_0^- and Q_1^- computed in Theorem 5.1. Let P_i be the only extreme point of Q_i^- $(i = 1, 2)$. Consider the following linear system:

$$\sum_{i=1}^{n} \alpha_i a_i = P_1 - P_0 = (\frac{1}{\pi_0} - \frac{1}{\pi_0 + 1})\pi \tag{26}$$

which has a unique solution $(\alpha_i)_{i=1\dots n}$. Then d defined by

$$P_0 + \sum_{i\ :\ \alpha_i \geqslant 0} \alpha_i a_i = P_1 - \sum_{i\ :\ \alpha_i < 0} \alpha_i a_i =: d$$

lies in $Q_0^- \cap Q_1^-$; therefore it is in Q^-, and it defines a cut. To compute it, it suffices to solve a linear system in dimension n. We will use this particular element of Q^- in section 6.2.

Remark This cut is the one used in [6] to estimate the quality of a branching direction.

6 Algorithm

The previous sections naturally suggest to compute split cuts by projecting the origin onto Q^-, decomposing the projection as a convex combination of extreme points, and using these extreme points of Q^- as cutting directions. But before using this approach, we need to deal with several issues.

- Theorem 2.2 fails when $lin(Q) \neq \mathbb{R}^n$.
- The characterization of the normal cone in Theorem 4.2 fails in general for an empty polyhedron.
- Algorithm 1 can only be used on a bounded polyhedron (a polytope); otherwise, the linear program in step 2 may be unbounded.
- According to (25), Q^- is described via variables $\alpha \in \mathbb{R}^{2m+2}$. If the LP in step 2 of algorithm 1 has several solutions, an extremal such solution in this extended space need not correspond to an extremal solution in $\mathbb{R}^n$.

6.1 When $lin(Q) \neq \mathbb{R}^n$

When $lin(Q) \neq \mathbb{R}^n$, Theorem 2.2 fails: facets of Q no longer correspond to extreme points of Q^-. Theoretically, this can be avoided by working in the Euclidean space $lin(Q)$ instead of $\mathbb{R}^n$. In the applications, we will ignore this problem and act as if $lin(Q)$ was equal to the whole space[1] $\mathbb{R}^n$. For this reason, Q^- may have no extreme point at all. However, thanks to the normalization constraint presented in the next subsection, we will never run into this case: Q^- will be replaced by a bounded set $\tilde{Q}^-$.

6.2 Normalization constraint

It is clear from its definition that Q^- is unbounded. Unfortunately, the black box used by the successive projection algorithm needs the poyhedron to be bounded, so we must somehow bound Q^- artificially to use our method. The description (25) suggests to bound Q^- by replacing it with $\tilde{Q}^-$, defined as

$$\tilde{Q}^- = \left\{ d = \sum_{j=1}^{m+1} \alpha_0^j a_0^j = \sum_{j=1}^{m+1} \alpha_1^j a_1^j,\ \alpha_i^j \geqslant 0,\ \sum_{i,j} \alpha_i^j \leqslant N \right\} \tag{27}$$

where N is a sufficiently large normalization constant. Note that it suffices to compute the particular cut described in section 5.3 to get a value of N (say, $N = \sum_i |\alpha_i|$ where α is the solution of the linear system (26)) ensuring that $\tilde{Q}^- \neq \emptyset$.

[1] Or $lin(Q) = \{A_e x = b_e\}$ if the problem has equality constraints $A_e x = b_e$.

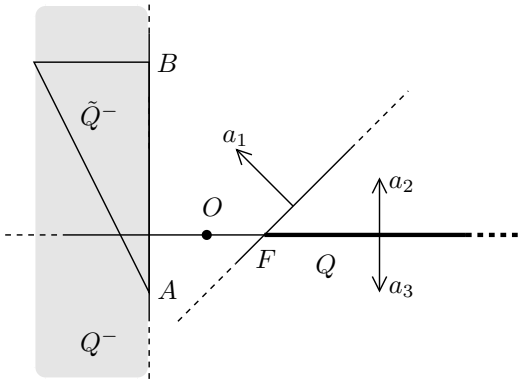

Figure 10: A degenerated example where $lin(Q) \neq \mathbb{R}^n$

Remark Replacing Q^- with $\tilde{Q}^-$ may introduce new extreme points which do not correspond to facets of Q. The corresponding cuts will therefore not be facet-defining. This seem to be unavoidable, since we have seen in Figure 5 that the deepest cut need not be implied by facet-defining cuts.

Figure 10 illustrates the problem raised by $lin(Q) \neq \mathbb{R}^n$ and the role of the normalization constraint. Q is defined by the constraints $a_i \cdot x \leqslant b_i$ $(i = 1, \ldots, 3)$ and $lin(Q)$ is the horizontal axis. The projection of O onto $\tilde{Q}^-$ is decomposed as a convex combination of points A and B, and two cuts are generated which expose the same facet F of Q.

6.3 Normal cone of an empty polyhedron

In this subsection, we show that Lemma 4.1 does not hold whithout the assumption that the polyhedron is nonempty. Let $n = 2$ and

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -1 \\ -1 \end{pmatrix}.$$

Then $Q := \{(x = (x_1, x_2) \in \mathbb{R}^2 \ ; \ Ax \leqslant b\} = \emptyset$. The set of $u \in \mathbb{R}^2_+$ such that $b \cdot u \leqslant 0$ is simply $\mathbb{R}^2_+$ and the set $A^\top u$ for such u is $\mathbb{R} \times \{0\}$. If Lemma 4.1 held, the normal cone of Q would be $\mathbb{R} \times \{0\}$, but since Q is empty, its normal cone is the whole space $\mathbb{R}^2$.

In our situation, $Q = \overline{conv}(Q_0 \cup Q_1)$ may be assumed nonempty (otherwise the whole problem makes little sense). However, either Q_0 or Q_1 could well be empty, an event which might be computationally expensive to check. Fortunately, this difficulty is minor. In fact, assume for example $Q_0 = \emptyset$, so that $Q^- = Q_1^-$. No matter which wrong set (call it $\bar{Q}_0^-$) is obtained via Lemma 4.1 and Theorem 4.9 for the reverse polar of Q_0, elements of $\bar{Q}_0^- \cap Q_1^-$ will lie in $Q_1^- = Q^-$: they will provide valid cuts for Q. In other words, in our application to integer programming, there is no risk to cut a feasible solution.

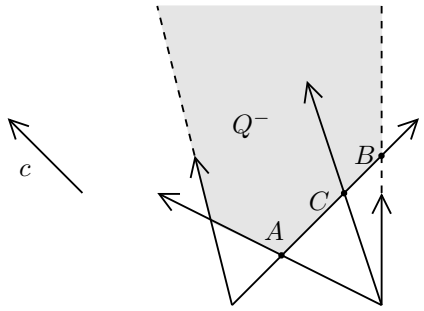

Figure 11: The point returned by the black box may not be extreme

6.4 The black box may return a non-extreme point

In algorithm 1, the oracle called in step 2 was supposed to return an extreme point of Q^-. However, in our application to disjunctive programming, this cannot be guaranteed. The situation of figure 11, where Q is the intersection of two cones as in (25), may happen: the whole line segment AB is optimal for the minimization of $x \mapsto c \cdot x$ over Q^-, and the points A, B and C (note that C is not extreme in Q^-) can all be returned by the solver which is provided the description (25) of Q^-.

This problem is not too serious however: it occurs in degenerate cases only (c has to be very particular) and if it does, the possible existence of a non-extreme point of Q^- in the decomposition of the projection of O onto Q^- corresponds to generating a cut which is not facet-defining, but at least it is valid.

6.5 Algorithm

Summing up, we suggest the following method to separate the origin from the closed convex hull of the union of two disjunctive polyhedra.

– Consider a mixed integer program (20) and obtain an extreme point, optimal solution $\overline{x}$ to the linear relaxation (21). Translate the origin O to $\overline{x}$.
– Choose a splitting direction i such that $\overline{x}_i$ is not integer although x_i is subject to integrality in (20). Split the polyhedron of the linear constraints in (20) into Q_0 and Q_1.
– Use Theorem 5.1 to build ${Q_0}^-$ and ${Q_1}^-$.
– Intersect ${Q_0}^-$ and ${Q_1}^-$ to get Q^-.
– Normalize Q^- into $\tilde{Q}^-$, for example with the help of section 5.3, and design a black box which minimizes over $\tilde{Q}^-$.
– Use the successive projection algorithm 1 to compute the projection of the origin O onto Q^- and its decomposition in extreme points $d_1, \ldots, d_k$ of Q^-.
– Use the constraints $d_i \cdot x \leqslant -1$, $i = 1, \ldots, k$ as cuts.

This cut generation algorithm was implemented and tested on small random instances of integer programs. The results are encouraging, but more numerical experiments are still needed to evaluate its practical interest. In particular, it would be interesting to embed such a cut generator in an existing branch-and-cut algorithm to see how it performs on classical problem libraries. This extensive numerical work is beyond the scope of this paper and is a natural direction for future work.

Conclusion

We aim at finding "good" hyperplanes separating O from a convex set Q. The reverse polar Q^- of Q determines cutting directions, and moreover, when Q is a polyhedron, its extreme points (up to intersection with a convenient subspace) determine cuts that expose a facet of Q. Besides, the problem of projecting O onto Q to find the deepest cut amounts to projecting it onto Q^-. As a consequence, we suggest to compute the projection of O onto Q^- along with its decomposition in extreme points of Q^-. We then use these points to generate several cuts that separate O from Q. This technique applies in particular to disjunctive programming, when the polyhedra are explicitly described by inequalities : it is then possible to do all the computations explicitly. This results in an implementable algorithm, the practical interest of which still needs to be studied.

Acknowledgement. The author wishes to thank his advisor Claude Lemaréchal for his help, comments and suggestions all along this research.

References

[1] E. Balas. Disjunctive programming. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II, 5*, pages 3–51, Amsterdam, 1979. Annals of Discrete Mathematics, North-Holland.

[2] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Prog.*, 58(3):295–324, 1993.

[3] G. Cornuéjols. Valid inequalities for mixed integer linear programs. *Math. Prog.*, 2006. To appear.

[4] G. Cornuéjols and C. Lemaréchal. A convex-analysis perspective on disjunctive cuts. *Math. Prog.*, 106:567 – 586, 2006.

[5] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer Verlag, Heidelberg, 2001.

[6] Miroslav Karamanov and Gérard Cornuéjols. Branching on general disjunctions (submitted). 2005.

[7] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. *Lecture Notes in Computer Science*, 2081:348–360, 2001.

[8] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley and Sons, 1986.

[9] Ph. Wolfe. Finding the nearest point in a polytope. *Math. Prog.*, 11:128 – 149, 1976.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399